Computer Architecture
ID: 519021911248
Name: ZhuoHao Li

Homework 2

2022-3-26

EX.1

(a)

assuming that 1 is for enable, 0 is for disenable

1: Control Signal

ALUop	ExtSel	ALUMux	MemR	MemW	RegW	RegDst	PCSrc	RegMux	Branch
`add`	*	Register	0	0	1	rd	PC + 4	1(ALU)	0

(b)

The following steps are performed: the instruction address is fetched from the PC, then the operands are fetched from the instruction memory, ALU is used to perform the operation, and the result is written to registers.

So **program counter**, **instruction memory**, **registers**, and **ALU** are the resources (blocks) that perform useful functions for this instruction.

(c)

Branch Add is the resource that produces output but is not used. Its output is not used for this AND instruction. It is used by the branch instructions to find the next instruction address.

Data Memory is the resource that does not produce any out in this AND instruction. It produces output for load and store instructions.

EX.2

(a)

The whole process is like this: Intstruction Fetch (PC and instruction memory), Decode (Register), Execute (ALU), Memory Read and Write (Data Memory).

So, **PC**, **Instruction Memory**, **Registers(a Read Port and a Write Port)**, **ALU**, **Data Memory** are the resources (blocks) that perform useful functions for the instructions.

(b)

We need to extend the existing ALU to also do **shifts** (SLL, to extend the offset to 32bit value).

(c)

We need to change the ALU operation control signals to support the SLL operation in the ALU.

EX.3

(a)

Both `lw` and `sw` utilizes the data memory. So the fraction can be calculated as sum of `lw` and `sw` , which is 25% + 10% = 35%

(b)

Except `add` and `not` all of then need sign extended circuit. So the fraction is sum of `addi`, `beq`, `lw` and `sw`

For example, `addi` will extend an immediate and have it with add operation. `beq`, will also handle sign extension when computing the next PC values.`lw` and `sw` will compute the target address of the data

waiting to be written or store.

Required answer(fraction) is 20% + 25% + 25% + 10% = 80%

The circuit just calculate the un-needed outputs. Control signals will filter those outputs so we actually don't care what the exact values of them.

EX.4

(a)

pipelined: it's determined by the longest path in a system (critical path). so **CT= 350ps**

non-pipelined: it's the value of the sum, which is **CT=250+350+150+300+200=1250ps**

(b)

pipelined: $T = 5 \times CT = 1750ps$, every instruction will be executed in **5** stages.

non-piplined: $T = 1250ps$, every instruction will be excuted in a complete single cycle. The value can be computed by summing stage time all .

(c)

In order to get the ideal bonus, we can split the longest path, which is $ID = 350ps$. After that, the new cycle time is $300ps$ from the new longest path MEM.

(d)

Data memory is used by `lw` and `sw`, so the utilization of data memory is 30% + 20% = 50%

(e)

Write-register port is used by `ALU`, `lw`, so the utilization od write-register port is 40% + 30% = 70%

(f)

Set pipelined as the merit standard.

For multi-cycle,

2: Multi-cycle Stages

`ALU`	4	no MEM
`beq`	4	no WB
`lw`	5	
`sw`	4	no WB

Multi-cycle Multi-cycle Execution time is X times pipelined execution time, where X is:

$$(5 \times 30\%) + (4 \times (40\% + 10\% + 20\%)) = 4.3$$

Single-cycle: Single-cycle Execution time is X times pipelined execution time, where X is:

$$\frac{CT_{single-cycle}}{CT_{pipelined}} = \frac{1250}{350} = 3.57$$